Integrating OWL Ontologies with a Dialogue Manager*

Universidad de Sevilla
{ gperez, jgabriel, pmanchon, fernando, jgonzalez}@us.es

Resumen: Este artículo describe la integración de ontologías en formato OWL como fuentes de conocimiento externas de un sistema de diálogo. El presente trabajo se centra en conseguir un agente que gestione cualquier ontología independientemente del dominio y sin perder expresividad al aprovechar los razonadores existentes sobre OWL y la estructura estática Sujeto-Propiedad-Objeto de RDFS común a todas las ontologías con formato OWL.
Palabras clave: Sistemas de gestión de diálogo, Ontologías.

Abstract: This paper describes the integration of OWL ontologies as external knowledge resources for dialogue systems. The current work focuses on implementing a domain-independent agent whose role is to deal with any ontology without losing expressivity, by using the existing OWL reasoners and the static structure Subject-Property-Object common to every ontology with OWL format
Keywords: Dialogue Systems, Ontologies.

1 Introduction

This paper describes the use of OWL ontologies as the external knowledge resource for a Dialogue Manager, focusing on reusability while maintaining the expressivity. The paper presents the advantages of our approach and describes how we have actually applied it within a real Dialogue Manager system: Delfos (Gabriel Amores, 2001). A description of the real ontology developed for the in-home domain is also included.

The separation of the knowledge management from the interaction modules has been argued in previous works. The main reason for this separation is that the knowledge management is conceptually different from the dialogue management and is not even necessary for all cases (Dahlback and Jonsson, 1999). In (Annika Flycht-Eriksson, 2000) the authors also point out that this separation facilitates the portability of dialogue systems to new domains. Today, a number of spoken and multimodal dialogue systems include a specific Knowledge Manager module, like Malin (Arne Jonsson, 2003), Javelin (Eric Nyberg, 2003) or Delfos (José Francisco Quesada, 2002). The agent described hereby preserves this separation between the Dialogue Manager and the Knowledge Manager.

Question-answering systems have traditionally studied the problem of linking the dialogue manager with the knowledge resources. In these systems the expected goal of the interaction with the users is to provide them with some information extracted from the knowledge resource. Therefore the knowledge management was often treated as an extension or a part of the dialogue manager and not as an independent module. In (Joseph Polifroni, 2002) the authors propose an extension to Galaxy (Stephanie Seneff, 1998) so that the Dialogue Manager becomes a domain-independent question-answering system, being able to automatically build the queries without changing the code, storing the domain-specific data in a table with standardized format. The approach presented in this paper keeps the idea of promoting portability by keeping domain independency, but extends the coverage of the solution: not limiting it to question-answering systems but extending it to any dialogue that may need reference resolution from external knowledge resources.
The solutions for data storage in dialogue systems that are proposed in nowadays systems can be classified in three groups: databases, ontologies and propietary solutions. There are arguments in favor and real implemented examples of each one of the approaches: see (Arne Jonsson, 2003) for a

* Part of this research has been funded by the Spanish Ministry of Education under Grant TIC2002-00526, and the European FP6 IST Talk Project (507802).

solution based on database access through SQL, (Iryna Gurevych, 2003) for arguments in favor of using OWL based ontologies or (Joseph Polifroni, 2002) for a table-based proprietary solution. The general discussion goes beyond the scope of this paper, but in section 3 there is a set of arguments in favor of the OWL approach for the scenario under study.

Summarizing, the solution presented in this paper consists on a independent module that offers knowledge management capabilities to the Dialogue Manager. The Knowledge Manager is both domain-independent: no need to change the module from one domain to another, and general-purpose: it works for question-answering systems but can also be used for reference resolution in any other dialogue. The knowledge resources are structured as OWL ontologies.

In section 2 an overall view of the system is presented. Section 3 introduces OWL and explains the advantages of this approach. In section 4 the actual ontology used for the In-Home domain is described. To wrap up, a set of real showcases (section 5) and the conclusions and future work (section 6).

2 Overall Architecture

The present work is included in the framework of a multimodal application in the In-Home domain. Although the results can be extrapolated to other user profiles, the current scenario is focused on wheel-chair bound users and their special circumstances. The scenario has been detailed after a set of WOZ experiments described in (Pilar Manchón, 2006).

In this particular scenario, users are able to access the system at all times through different modalities, that is, using speech and/or a graphical interface. The scenario includes microphones, speakers and a touch screen where the information can be displayed and introduced or selected.

The software components of the system are implemented as independent OAA (Open Agent Architecture) agents (David Martin, 1999), linked therefore through the OAA facilitator. An overall view of the system is shown in figure 1.

The core of the System is the Dialogue Manager agent, whose role is to control the course of the interactions with the user, checking the Multimodal Input Pool for

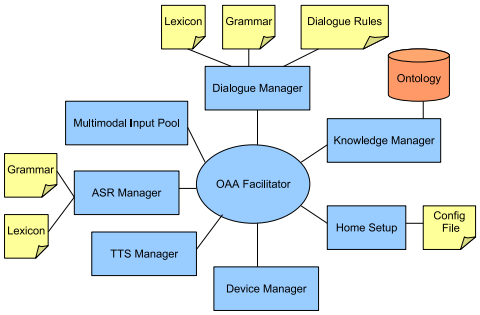

Figure 1: Overall Architecture

new inputs from the user, that may come from the ASR agent (speech) or from the Home Setup agent (clicks). The presentation of information from the Dialogue Manager to the user can also be done multimodally: by voice through the TTS agent and graphically through the Home Setup agent. The Dialogue Manager is inspired on the Information State Update approach (David Traum, 1999), and includes Delfos (Gabriel Amores, 2001) at dialogue level and Episteme (Gabriel Amores, 1997) for Natural Language Understanding.

The Dialogue Manager uses the Knowledge Manager described in this paper for resolving references to particular devices, that may be referred to by its label but also by its location, type, etc.

Finally the Dialogue Manager may decide to execute commands (e.g. switch on a light) using the x10 protocol through the Device Manager.

A detailed description of the scenario and the role and services offered by each agent can be found in (Tilman Beckerm, 2006).

3 Extending the Knowledge Manager

3.1 OWL

3.1.1 OWL Overview

OWL is the Web Ontology Language recommendation from the W3C, and has been defined to be compatible with the architecture or the World Wide Web in general, and the Semantic Web in particular. The recommendation is built on top of RDF:

"OWL builds on RDF and RDF Schema and adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes" (http://www.w3.org/2004/OWL/). OWL adds the following capabilities to ontologies:

- Scalability to Web needs
- Compatibility with Web standards for accessibility and internationalization
- Possibility of constructing classes
- Restrictions on properties

And it has other generic advantages:

- It has been defined as a standard by W3C
- A big number of free resources available for development (APIs, Editors, Visualizators, etc.)

3.1.2 OWL Layers

OWL was divided into layers to reflect compromises between expressability and implementability. Three layers of OWL are defined: Lite, DL, and Full, in increasing level of expressiveness:

- *OWL Lite* supports those users primarily needing a classification hierarchy and simple constraints.(http://www.w3.org/TR/owl-ref/#OWLLite).
  - For example: it only permits cardinality values of 0 or 1
  - It also has a lower formal complexity than OWL DL.
- *OWL DL* : maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs under certain restrictions. (http://www.w3.org/TR/owl-ref/#OWLDL)
- *OWL Full* : maximum expressiveness and the syntactic freedom of RDF with no computational guarantees.(http://www.w3.org/TR/owl-ref/#OWLFULL)

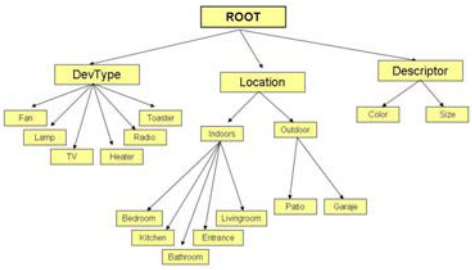

Figure 2: Former USE Ontology

3.2 Motivation

In former versions or our system, a naïve ontology manager was implemented (José Francisco Quesada, 2002) , which allowed to define semantic graphs of a house domain. These graphs looked like the tree shown in figure 2.

In order to work with such graphs, a Knowledge Manager was built as an OAA agent that solved reference resolution queries over tree–shaped graphs. The goal of each query was to identify one or several unknown devices by means of a list of "positive" and "negative" filters. That is, it specified the values which should be satisfied (or not) by the device (or devices) within the graph (color, device type, etc.). For instance, given the user request:

User Could you please turn on all the big lamps except the one in the kitchen?

The Knowledge Manager would solve a query such as:

```
KMDevRes(Pos[size:big,devtype:lamp],
   Neg[location:kitchen])
```

As shown in this example, the Knowledge Manager enables the system to have a flexible interaction with the user using natural language commands and reference resolution. Nevertheless, this agent could be improved in a number of ways, by:

1. Not imposing tree structures to the semantic network

2. Allowing other relationships between concepts (not just subsumption). For instance, a *television* is *locatedIn* a *room*, a *lamp hasColor red*, etc.
3. Allowing the definition of property features. For example transitivity over the property *locatedIn*: (if **Lamp** is *locatedIn* **Kitchen** and **Kitchen** is *locatedIn* **House**, then **Lamp** is *locatedIn* **House**).
4. Allowing the definition of restrictions over properties. For instance, a given **Device** must be described by one (and only one) **Color**.
5. Having a more generic ontology interface, which allowed abstract queries independent from the particular knowledge domain.

As the current needs were analyzed and some research on current standards and tools was carried out, we realized that the already existing standard OWL in conjunction with the querying language RDQL (RDF Data Query Language) and the reasoners included within the open–source Jena platform could help a great deal:

1. Since OWL is built on RDF, the semantic network could be defined by subject–property–object triplets without any predefined hierarchy and, therefore, no imposition of a tree structure.
2. OWL includes subsumption (`SubPropertyOf`, `SubClassOf`), but they are just two alternative possible relations. Actually, any subject–object pair can be linked by any user–defined property.
3. The reasoners included within Jena allow the definition of inference rules for all layers.
4. In OWL (Sub)classes can be created by restricting a property's behavior on that class.
5. RDQL provides a way of specifying a graph pattern that is matched against the graph to yield a set of matches. This approach provides a very flexible interface with the dialogue manager, since it can generate queries using any graph pattern.

Once it was determined that using OWL, Jena and RDQL the intended objectives could be achieved, an independent agent had to be implemented to substitute the previous Knowledge Manager. For this agent to comply with the goals described in section 1, the agent had to offer:

Reusability: It should provide a mechanism to make abstract (domain independent) queries, useful for any ontology.

Expressivity: The agent should allow the dialogue manager to generate queries that might turn out to be necessary during the dialogue.

Both requisites were fulfilled by implementing two different OAA *solvables*:

- One in charge of queries about the *domain* of the property.
- A second one in charge of queries about its *range*.

This approach is completely independent of the particular ontology used and therefore reusable in any domain (only limited by the expressivity of the queries).

Since the knowledge management is implemented as a separated agent, the replacement did not imply changes over other components of the system, like modifying the dialogue strategies or retraining the speech recognizer.

3.3 Implementation

The concrete implementation of the Knowledge Manager offers two services:

1. DQ: Queries about the domain (Subjects) of some properties.
   (a) Parameters:
      i. P1: The class searched for.
      ii. P2: A list of `[nameProperty, value]` pairs; that is, of properties which the subject must satisfy.
      iii. P3: a list of `[nameProperty, value]` pairs; that is, of properties which the object must not satisfy.
   (b) Solution: The individuals belonging to Class P1 and compatible with the graph restrictions defined in P2 and P3.

2. RQ: Queries about the range (Objects) of some properties, with the following parameters:
   (a) Parameters:
      i. P1: The list of individuals searched for.
      ii. P2: A list of properties.
   (b) Solution: All the individuals that are linked to any of the individuals listed in P1 by any of the properties listed in P2.

To elaborate further on how these queries are translated into RDQL and the results obtained, consider the first of the two services (i.e. Domain Queries). First, the agent looks for triplets whose object belongs to the class specified by P1 (i.e. `NameClass`), which is expressed as `NameClass:(?obj rdf:type NS:NameClass)` in RDQL syntax.

Then, it looks for the group of individuals (group "A", below), which satisfies the P2 restriction:

```
SELECT ?obj WHERE (?obj rdf:type
NS:NameClass)
(?obj NS:prop1 NS:value1)
...
(?obj NS:propN NS:valueN)
→ Result A
```

The next step is to find the set of M individuals which satisfy the P3 restriction:

```
SELECT ?obj WHERE (?obj rdf:type
NS:NameClass)
(?obj NS:prop1 NS:value1)
→ Result B1
...

SELECT ?obj WHERE (?obj rdf:type
NS:NameClass)
(?obj NS:propM NS:valueM)
→ Result BM
```

Considering that the group B is defined as the union of the individuals B_i:

$$ResultB = B1 \cup B2 \cup \ldots \cup BM$$

the final result is:

$$Result = ResultA - (ResultA \cap ResultB).$$

The actual wrappers are available for the general public at www.us.es/julietta/tools.html

4 Ontology for the In-Home Domain

4.1 Developing a Home Ontology in OWL

Once it was determined that OWL would be the standard chosen, Protégé 3.1 open source ontology editor was chosen to help the development of the ontology. Protégé is extensible and based on Java, and allows users to construct domain ontologies in various formats such as OWL, RDF, XML and HTML (http://protege.stanford.edu).

4.1.1 Ontology Coverage

An additional objective within this task was the design of a more exhaustive and complete version of the home ontology, which should include the "telephone operator" functionality defined in Siridus (Siridus Project). For this purpose, new devices and relations have been included and a new ontological structure has been designed. Obviously, the ontology described below is just for illustration purposes, and is not meant to be exhaustive, neither in its class coverage nor in the instantiation of individuals.

4.1.2 Classes and Subclasses

The basic element in OWL consists of the triplet Subject–Predicate–Object. Subjects and objects are denoted by classes and subclasses, while Predicates are typically denoted by properties.

System: is a special class, used to describe the role of the dialogue system in the ontology and how it interacts with the objects of this universe.

Device: Describes device types, and contains subclasses like:

1. **Lamp** and **Dimmer** (actually a subclass of **Lamp**)
   (a) *Lamp_1* through *Lamp_6* as individuals
   (b) *DimmerLamp_1* and *DimmerLamp_2* individuals
2. **Blind** (one individual, *Blind_1*)
3. **Radio** (one individual, *Radio_1*)
4. **TV** (one individual, *TV_1*)

Area includes the following individuals: *Upstairs*, *Downstairs*, *Indoors*, *Outdoors*

Room includes other subclasses like:

1. **Bathroom** (*Bathroom_1* and *Bathroom_2*)
2. **Bedroom** (*Bedroom_1* through *Bedroom_4*)
3. **Garage** (*Garage_1*)

SpecificLocation contains 5 sample individuals: *Ceiling* , *LeftCorner* , *Table* , *Wall* , *RightCorner*

Color is a class available for both rooms and devices, whose individuals are: *Black*, *Blue*, *Green*, *Red*, *White*, *Yellow*.

Size has two individuals: *Big*, *Small*.

The actual functionality taken into account in this new version of the ontology includes the following commands for the home devices:

- On/Off, Open/Close, Raise/Lower

As discussed previously, the telephone operator functionality and its corresponding directory of entries have also been integrated in the new ontology. However, they deserve a special consideration within the ontology with respect to the rest of the devices and/or functionalities. Ontologically, a class of name **Directory** describes the directories available in the home (currently, it only contains one individual).

The information stored in each entry in the directory is defined as a **DirectoryEntry** class, with six individual instantiations in our example (*DirectoryEntry_1* to *DirectoryEntry_6*). Each Directory Entry requires a set of objects, defined as independent classes:

- **FirstName** (several instances)
- **LastName** (several instances)
- **HomeNumber** (no instances)
- **Mobile** (no instances)
- **Email** (20 individuals)
- **Relationship** to the user, with the following individuals: *Boss*, *Brother*, *Cousin*, *Father*, *Friend*, *Mother*, *Neighbour*, *Sister*, *Uncle*

4.1.3 Properties

Instances of objects are linked to other objects according to the OWL triplet by means of properties and subproperties. The hierarchy of properties developed for the In-Home domain is as follows:

The property **hasDeviceCommand** is conceptualized as a **System** class performing a set of functions or commands over the set of devices:

- **hasDeviceCommand** (System hasDeviceCommand Device)
  - **Domain**: System, **Range**: Device

This property contains a series of subproperties, each corresponding to one type of functionality over the devices in the house:

- **SwitchOn** and **SwitchOff**, whose ranges are the classes **Fan**, **Heater**, **Lamp** and **Radio**
- **Open**, **Close**, **Raise** and **Lower**, whose range is the class **Blind**

Additional properties have been defined for the **Device** class, which are not related to the devices' functionality, such as:

- **hasColor** (Device hasColor Color)
  - **Domain**: Device, **Range**: Color
- **hasSize** (Device hasSize Size)
  - **Domain**: Device, **Range**: Size
- **locatedIn** a transitive function with four domains and ranges
  - **Domain**: Device, SpecificLocation, Room, Area. **Range**: SpecificLocation, Room, Area, Home (respectively)

The functionality considered for the telephone operator is described by the subproperties of the **hasTelephoneCommand** property. All of them take **System** as their Domain, and **DirectoryEntry** as their **Range**:

- **Call, MakeConference, Transfer, CancelTransfer, List, Find, Redial**

Additional properties have also been defined for the **DirectoryEntry** class which are not related to its functionality, such as:

- **hasEmail** (DirectoryEntry hasEmail Email)
  - **Domain**: DirectoryEntry, **Range**: Email

. . . and so on for each property: **hasFName**, **hasHName**, **hasHNumber**, **hasLName**, **hasNNumber**, **hasRelationship**, **hasSize**, **hasWNumber**.

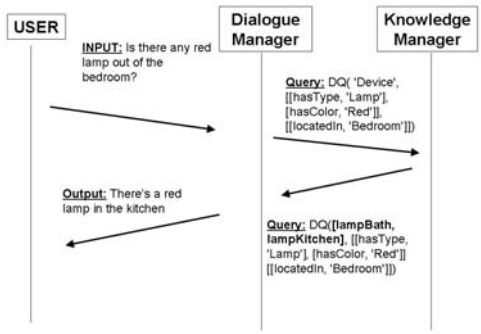

Figure 3: Direct User Query 1

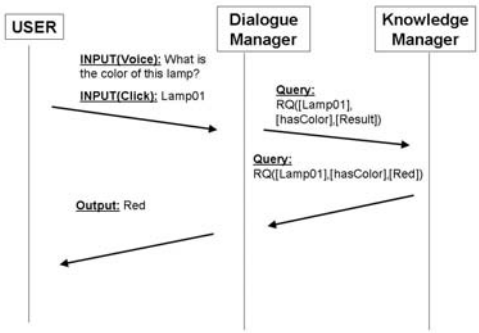

Figure 4: Direct User Query 2

5 *ShowCases*

In this section three real examples are presented to show how the Dialogue Manager interacts with the Knowledge Manager. The first two examples show how the Knowledge Manager can be applied in the context of a question-answering dialogue. The third example shows how the same agent is used by the Dialogue Manager for reference resolution in a general-purpose dialogue.

ShowCase 1: Direct User Query This example shows how the Dialogue Manager would translate a direct query from the user. The user is interested in knowing which devices satisfy a set of conditions. The situation is depicted in figure 3.

ShowCase 2: Direct user Query Again, this example shows how the Dialogue Manager would translate a direct query from the user, but in this case the question is related to the properties of a known device, as exemplified in figure 4:

ShowCase 3: Reference resolution (underspecification) Next, a more sophisticated situation is described. The user gives a command so the Dialogue Manager asks the Knowledge Manager which device the User is referring to. The result of this query implies that the command was ambiguous, so the Dialogue Manager asks the User for more details. The scenario is shown in figure 5. This functionality is implemented as an expectation of the Dialogue Manager (José Francisco Quesada, 2000).

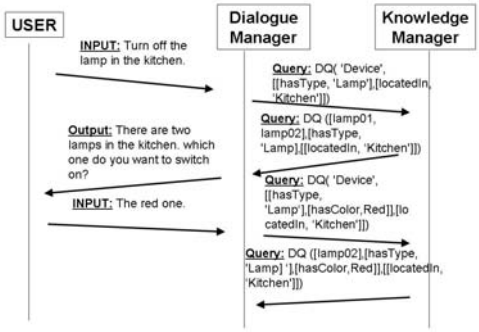

Figure 5: Reference Resolution

6 *Conclusions and future work*

This paper describes a proposal for integration of OWL ontologies within dialogue systems. A domain-independent OAA agent is described, allowing its use for any ontology, easing therefore the portability of the system. Its integration with a real Dialogue Manager has been shown using Delfos and an In-Home ontology has been detailed. The paper also includes real showcases that exemplify the use of the agent both for question-answering and for reference resolution.

In addition of the implementation of this agent, a tool to generate grammars from OWL ontologies has been implemented, which ensures the coherence between the Knowledge resource and the NLU (Natural Language Understanding) module of the Di-

alogue Manager. A detailed description of this tool can be found in (Guillermo Pérez, 2006).

In later versions of the Knowledge Manager, some improvements will be included: mainly extending the coverage of OWL layers (now limited to Lite) and augmenting the expressivity by allowing filters for the Range queries. Other steps to be taken include migrating from RDQL to the newer SPARQL (www.w3.org/TR/rdf-sparql-query/) and checking our approach with other pre-defined ontologies.

References

[Annika Flycht-Eriksson2000] Annika Flycht-Eriksson, Arne Jonsson. 2000. Dialogue and domain knowledge management in dialogue systems. In *1st SIGdial workshop on Discourse and dialogue*, october.

[Arne Jonsson2003] Arne Jonsson, Magnus Merkel. 2003. Some issues in dialogue-based question-answering. In *Working Notes from AAAI Spring Symposium.*

[Dahlback and Jonsson1999] Dahlback, Nils and Arne Jonsson. 1999. Knowledge sources in spoken dialogue systems. In *Eurospeech '99.*

[David Martin1999] David Martin, Adam Cheyer, Douglas Moran. 1999. The open agent architecture: A framework for building distributed software systems. 13:91–128.

[David Traum1999] David Traum, Johan Bos, Robin Cooper Staffan Larsson Ian Lewin Colin Matheson Massimo Poesio. 1999. A model of dialogue moves and information state revision. Deliverable 2.1, Trindi Project.

[Eric Nyberg2003] Eric Nyberg, Teruko Mitamura, James P. Callan Jaime G. Carbonell Robert E. Frederking Kevyn Collins-Thompson. 2003. The javelin question-answering system at trec 2003: A multi-strategy approach with dynamic planning. In *Twelfth Text REtrieval Conference.*

[Gabriel Amores1997] Gabriel Amores, José Francisco Quesada. 1997. Episteme. *Procesamiento del Lenguaje Natural*, 21:1–16.

[Gabriel Amores2001] Gabriel Amores, José Francisco Quesada. 2001. Dialogue moves for natural command languages. *Procesamiento del Lenguaje Natural*, 27:89–96.

[Guillermo Pérez2006] Guillermo Pérez, Gabriel Amores, Pilar Manchón David González. 2006. Generating multilingual grammars from owl ontologies.

[Iryna Gurevych2003] Iryna Gurevych, Robert Porzel, Elena Slinko Norbert Pfleger Jan Alexandersson Stefan Merten. 2003. Less is more: using a single knowledge representation in dialogue systems. In *Human Language Technology Conference, HLT-NAACL 2003 workshop on Text meaning*, pages 14–21.

[José Francisco Quesada2000] José Francisco Quesada, Gabriel Amores. 2000. Dialogue moves in natural command languages. Deliverable 1.1, Siridus Project.

[José Francisco Quesada2002] José Francisco Quesada, Gabriel Amores. 2002. Knowledge–based reference resolution for dialogue management in a home domain environment. In Mary Ellen Johan Bos and Colin Matheson, editors, *Proceedings of the sixth workshop on the semantics and pragmatics of dialogue (Edilog)*, pages 149–154, 4-6th September.

[Joseph Polifroni2002] Joseph Polifroni, Grace Chung. 2002. Promoting portability in dialogue management. In *ICSLP2002.*

[Pilar Manchón2006] Pilar Manchón, Carmen del Solar, Gabriel Amores Guillermo Pérez. 2006. In *Generating Multilingual Grammars from OWL Ontologies.*

[Stephanie Seneff1998] Stephanie Seneff, Ed Hurley, Raymonf Lau Christine Pao Philipp Schmid Victor Zue. 1998. Galaxy-ii: A reference architecture for conversational system development. In *ICSLP1998.*

[Tilman Beckerm2006] Tilman Beckerm, Peter Poller, Nate Blaylock Staffan Larsson Oliver Lemon Guillermo Pérez Jan Schehl. 2006. Talk status report: Software infrastructure. Deliverable 5.1s2, Talk Project.